# Driving a Seven Segment Display with the NEURON® CHIP

August 1991 **LONWORKS™ Engineering Bulletin**

## Introduction

This engineering bulletin describes how the Echelon NEURON CHIP can be used to drive a seven-segment display controller chip, the Motorola MC14489. The MC14489 can control up to five LED digits, each consisting of seven segments and a decimal point. No external current limiting resistors or drive transistors are required. The chip has a Serial Peripheral Interface (SPI), allowing for easy connection to the NEURON CHIP's Neurowire port. This port can drive devices conforming to Motorola's SPI device interface and National Semiconductor's Microwire™ device interface. The engineering bulletin also presents software drivers written in the NEURON C programming language that display decimal numbers from binary data.

## Schematic

The MC14489 can be connected to the NEURON 3150™ CHIP or the NEURON 3120™ CHIP as indicated in the following schematic. The NEURON CHIP's Neurowire port uses pin IO_8 as the clock pin, and IO_9 as the serial output data pin. In this case, pin IO_2 is used as the enable pin for the MC14489 display controller, but any of pins IO_0 through IO_7 could have been chosen, with the appropriate modification to the driver software.

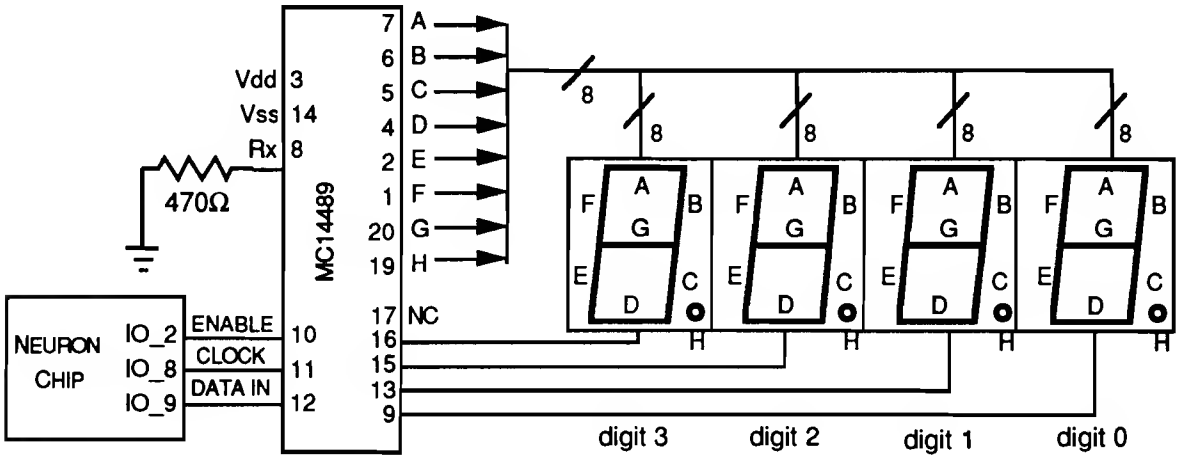



**Figure 1: Seven-segment LEDs controlled by the NEURON CHIP**

The MC14489 is connected to four common-cathode seven-segment LED display devices. These are available from most manufacturers of opto-electronic devices,

such as General Instruments, Hewlett-Packard, Industrial Electronic Engineers and William J. Purdy. The value of the current-limiting resistor connected between the Rx pin and ground depends on the application. If more than five digits are desired, several MC14489 devices may be connected in a cascade configuration, with the serial data being shifted out of one device into the next. See the Motorola MC14489 data sheet for more details. Other SPI or Microwire devices may be connected at the same time to the NEURON CHIP'S Neurowire port, provided each device has its own Enable pin.

## Programming

The MC14489 has two write-only device registers controlled by the software on the NEURON CHIP. The eight-bit configuration register shown in figure 2 contains bits that affect the decoding of the data in the 24-bit display register.

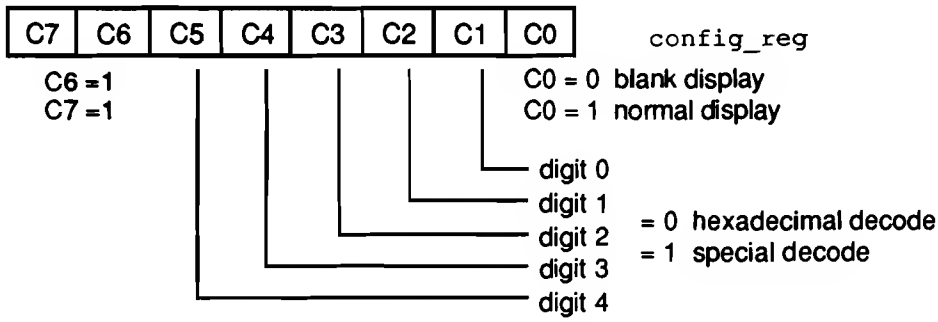
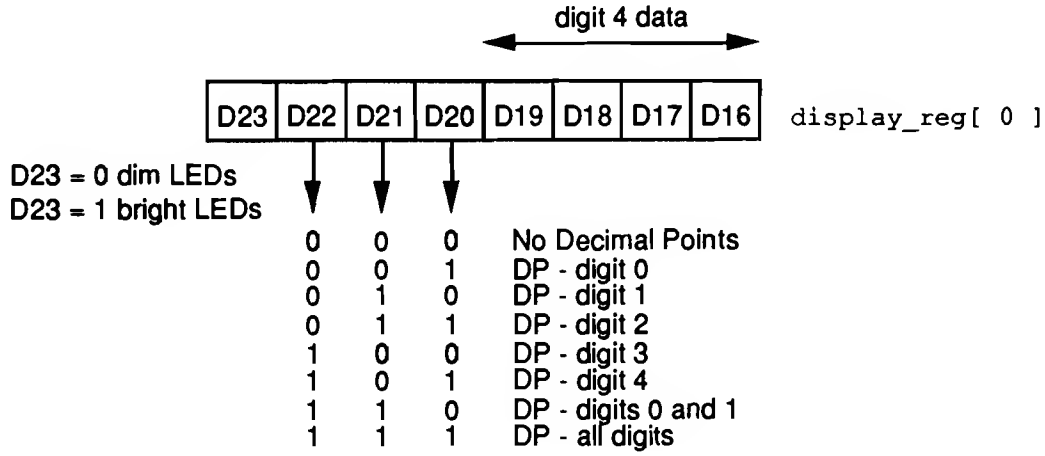



**Figure 2: MC 14489 configuration register**

The display register contains bits that define the display pattern of the LED digits and the decimal points as defined in figure 3.



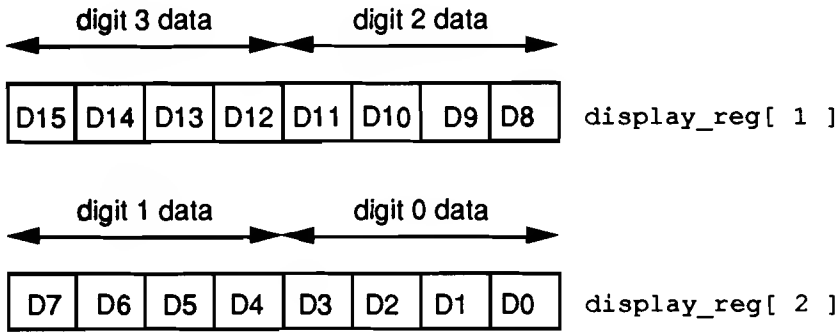



**Figure 3: MC14489 display register**

For the purposes of this application, there are two modes in which the data can be displayed. In hexadecimal mode, the four bits of data displayed in each digit are decoded as the hexadecimal digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. In special mode, certain other characters such as a space and a minus sign can be displayed. The table in figure 4 shows the patterns displayed in the two modes for all possible values of the data in the digit.

| digit data | hex decode | special decode |
|---|---|---|
| 0 | 0 | |
| 1 | 1 | r |
| 2 | 2 | H |
| 3 | 3 | h |
| 4 | 4 | J |
| 5 | 5 | L |
| 6 | 6 | n |
| 7 | 7 | o |
| 8 | 8 | P |
| 9 | 9 | r |
| A | A | U |
| B | b | u |
| C | C | y |
| D | d | - |
| E | E | = |
| F | F | □ |

**Figure 4: MC14489 display decoding**

The software uses the Neurowire (SPI) function of the NEURON C programming language to write data to these two registers. When the application program issues an `io_out` function call to the Neurowire device, the system software activates the chip select pin (in this case IO_2), and then clocks the data out on pin IO_9, using pin

IO_8 for the clock. The default rate of this serial data clock is 20 Kbit/sec when the NEURON CHIP input clock is 10 MHz. When eight bits of data are clocked into the MC14489, these data bits are written to the configuration register. When 24 bits of data are clocked in, they are written to the display register. The software drivers presented in listings 1 and 2 always write both the configuration and the display register sequentially.

## Software

Two listings are presented here. The first listing is for a simple decimal display function for positive numbers. Leading zeroes are not suppressed, so that, for example, the number 123 is displayed as 0123. There is also no error check for numbers that are out of range of the display. The built-in NEURON C library function `bin2bcd( )` performs the actual data conversion.

The function `display_number( number, dp_digit )` displays unsigned decimal numbers with a decimal point to the right of the specified digit. Note that the special values `NO_DP` and `ALL_DPS` may be used as the digit number to illuminate none or all, respectively, of the decimal points.

**Listing 1: Seven-segment display driver for positive numbers**

```
////////////////// SEVEN SEGMENT DISPLAY DRIVER  ////////////////////

// This NEURON C™ #include file contains code to drive the Motorola MC14489
// seven-segment display controller chip, interfaced to the NEURON® CHIP
// using Neurowire output mode.

// Pin IO_8 is the Neurowire clock, pin IO_9 is the serial output data
// Pin IO_2 is the chip select (may be modified).

// The function display_number( ) displays unsigned numbers
// with leading zeroes

////////////////////// DECLARATIONS ////////////////////////////////

IO_8 neurowire select( IO_2 ) IO_seven_seg;
IO_2 output bit IO_7_seg_select = 1;

struct bcd display_reg;              // 24 bits for 7-seg display reg
unsigned char config_reg;            //  8 bits for 7-seg  config reg

//////////////////////////// DISPLAY DECIMAL NUMBER ////////////////////

void display_number( unsigned long number, int dp_digit )
{
    config_reg = 0xc1;                              // Decimal decode all digits
    bin2bcd( number, &display_reg );                // Convert binary to decimal
    display_reg.d1 = 0x80 + dp_digit + 1;           // Set MS nibble for dec. pt.
    io_out( IO_seven_seg, &config_reg, 8 );         // Update device registers
    io_out( IO_seven_seg, &display_reg, 24 );
}

#define NO_DP   -1
#define ALL_DPS  6
```

The second listing is for a more user-friendly decimal display. It handles positive, negative and out-of-range numbers, and it also suppresses leading zeroes where appropriate.

The software functions are divided into three groups; low-level, character-oriented, and high-level.

### Low-level functions

The first group of functions provides low-level access to the display controller chip. The function `clear_display( )` clears a RAM copy of the configuration and display registers to a state that displays all blank characters. It does this by setting all digits to special decode mode, and writing the data for the blank character to all digits. The function `shift_out( )` uses the Neurowire device to write the contents of the RAM copy of the configuration and display registers to the actual MC14489 device registers.

### Character-oriented functions

The second group of functions are character-oriented routines that write into the RAM copy of the configuration and display registers. Note that they do not update the hardware device registers. The routine `shift_out( )` is used whenever this is desired.

The function `display_decimal( digit_number, decimal )` writes the specified decimal (0 - 9) into the specified digit position in the RAM copy of the display register. Digits are numbered from right to left, with digit 0 being the rightmost (units) digit. The software changes the decode mode for the specified digit to hexadecimal, and writes the appropriate data nibble into the RAM copy of the display register.

The function `display_DP( digit_number )` writes the appropriate bit in the RAM copy of the display register to illuminate the decimal point to the right of the specified digit.

The function `display_minus( digit_number )` writes the appropriate values in the RAM copy of the display register to illuminate the minus sign (segment G) in the specified digit.

### High-level functions

The third group of functions forms complete display images and updates the display hardware.

The function `display_OFF( )` causes the characters "OFF" to be displayed in the rightmost three digits. This is an example of a class of routines that display other alphanumeric messages composed of the limited character set available with seven segments per character. The letters available in upper case are "A, B, C, D, E, F, H, I, J,

L, O, P, S, U, Y, Z", and in lower case "c, d, g, h, i, l, n, o, r, u". If other letters, or more elegant letters are desired, an alphanumeric display should be used instead of a seven-segment display.

The function `display_number( number, dp_digit )` displays positive or negative decimal numbers with a decimal point to the right of the specified digit, with suppression of leading zeroes. Note that the special values `NO_DP` and `ALL_DPS` may be used as the digit number to illuminate none or all, respectively, of the decimal points. For a 4-digit display, the range of input numbers is from -999 to 9999. Numbers outside this range display as "----".

Table 1 shows some examples of the display produced by different input values with a four-digit display.

```
display_number( 123, 0 )         =>     1 2 3.
display_number( 123, 1 )         =>     1 2.3
display_number( 123, 2 )         =>     1.2 3
display_number( 123, 3 )         =>   0.1 2 3
display_number( 123, NO_DP )     =>     1 2 3
display_number( 123, ALL_DPS )   =>    .1.2.3.

display_number( -123, 0 )        =>   - 1 2 3.
display_number( -123, 1 )        =>   - 1 2.3
display_number( -123, 2 )        =>   - 1.2 3
display_number( -123, 3 )        =>   -.1 2 3
display_number( -123, NO_DP )    =>   - 1 2 3
display_number( -123, ALL_DPS )  =>   -.1.2.3.
```

**Table 1: Display produced by various input parameters to the display_number function.**

If it is desired to use this software with a different number of digits of display hardware, the defined symbols `NUM_DIGITS`, `MIN_NUMBER` and `MAX_NUMBER` in listing 2 should be changed according to table 2.

| NUM DIGITS | MIN NUMBER | MAX NUMBER |
|---|---|---|
| 1 | -9 | 99 |
| 2 | -99 | 999 |
| 3 | -999 | 9999 |
| 4 | -9999 | 32767 |

**Table 2: MIN_NUMBER and MAX_NUMBER values for different numbers of digits**

**Listing 2: General-purpose seven-segment display driver**

```
////////////// SEVEN SEGMENT DISPLAY DRIVER ////////////////////////

// This NEURON C™ #include file contains code to drive the Motorola MC14489
// seven-segment display controller chip, interfaced to the NEURON® CHIP
// using Neurowire output mode.

// Pin IO_8 is the Neurowire clock, pin IO_9 is the serial output data
// Pin IO_2 is the chip select (may be modified).

// Change the following #defines if there are fewer than 4 digits
// in the display. The rightmost digit is numbered 0.

#define NUM_DIGITS     4
#define MAX_NUMBER  9999
#define MIN_NUMBER  -999

////////////////////// DECLARATIONS ////////////////////////////////

IO_8 neurowire select( IO_2 ) IO_seven_seg;
IO_2 output bit IO_7_seg_select = 1;    // Initially unselected

unsigned char display_reg[ 3 ];         // 24 bits for 7-seg display reg
unsigned char config_reg;               //  8 bits for 7-seg  config reg

////////////////////// CONTROL DISPLAY HARDWARE ///////////////////////

void shift_out( void ) {        // update device hardware registers
    io_out( IO_seven_seg, &config_reg, 8 );
    io_out( IO_seven_seg, display_reg, 24 );
}

void clear_display( void ) {    // clear image of device registers
    display_reg[ 0 ] = 0x80;    // max brightness
    display_reg[ 1 ] = 0;       // blanks on all digits
    display_reg[ 2 ] = 0;
    config_reg = 0xFF;          // special decode on banks 1-5, normal mode
}

////////////////// SINGLE-CHARACTER DISPLAY FUNCTIONS //////////////

void display_decimal( int digit_number, int decimal ) {
        // display decimal number ( 0 - 9 ) on a digit

    display_reg[ 2 - digit_number / 2 ] |=
        ( digit_number & 1 ) ? ( decimal << 4 ) : decimal;
    config_reg &= ~( 1 << ( digit_number + 1 ) );
}

void display_DP( int digit_number ) {     // display decimal point on a digit
    display_reg[ 0 ] |= ( digit_number + 1 ) << 4;
}
```

```
void display_minus( int digit_number ) {  // display minus sign on a digit
    display_reg[ 2 - digit_number / 2 ] |=
        ( digit_number & 1 ) ? 0xD0 : 0x0D;
}

////////////////////////// DISPLAY 'OFF' TEXT STRING /////////////////

void display_OFF( void ) {       // display the text " OFF"
    display_reg[ 0 ] = 0x80;     // max brightness, no decimal points
    display_reg[ 1 ] = 0x00;     // ` O'
    display_reg[ 2 ] = 0xFF;     // `FF'
    config_reg = 0xD1;           // hex decode on banks 1-3, normal mode
    shift_out( );                // special decode on bank 4
}

////////////////////////// DISPLAY DECIMAL NUMBER ////////////////////

void display_number( long number, int dp_digit );

#define NO_DP   -1       /* display number without  decimal point  */
#define ALL_DPS -2       /* display number with all decimal points */

void display_number( long number, int dp_digit )
{
    int digit;
    long local_num;

    clear_display( );        // clear image of display registers

    if( ( number > MAX_NUMBER ) || ( number < MIN_NUMBER ) ) {
        display_reg[ 1 ] = display_reg[ 2 ] = 0xDD;
        shift_out( );        // update hardware
        return;              // display "----" for overrange
    }


    if( number < 0 ) {
        display_minus( NUM_DIGITS - 1 );         // leading minus sign
        local_num = - number;
    } else local_num = number;

    for( digit = 0;
      local_num || ( digit <= max( dp_digit, 0 ) ); digit++ ) {
                // convert binary to decimal with leading zero suppress
        if( ( number < 0 ) && ( digit == NUM_DIGITS - 1 ) ) break;
        display_decimal( digit, ( int )( local_num % 10 ) );
        local_num /= 10;
    }
    display_DP( dp_digit );  // display decimal point
    shift_out( ) ;           // update hardware
}
```

**Disclaimer**

Echelon Corporation assumes no responsibility for any errors contained herein.
No part of this document may be reproduced, translated, or transmitted in any form without permission from Echelon.



Echelon Corporation
4015 Miranda Avenue
Palo Alto, CA 94304
Telephone (415) 855-7400
Fax (415) 856-6153

Echelon Europe Ltd
105 Heath Street
London NW3 6SS
England
Telephone (071) 431-1600
Fax (071) 794-0532
International Telephone + 44 71 431-1600
International Fax + 44 71 794-0532

Echelon Japan K.K.
AIOS Gotanda Building #808
10-7, Higashi-Gotanda 1-chome,
Shinagawa-ku, Tokyo 141, Japan
Telephone (03) 3440-8638
Fax (03) 3440-8639

Part Number 005-0014-01